# CSc 174 Database Management Systems

## 4. EER to Relational Mapping

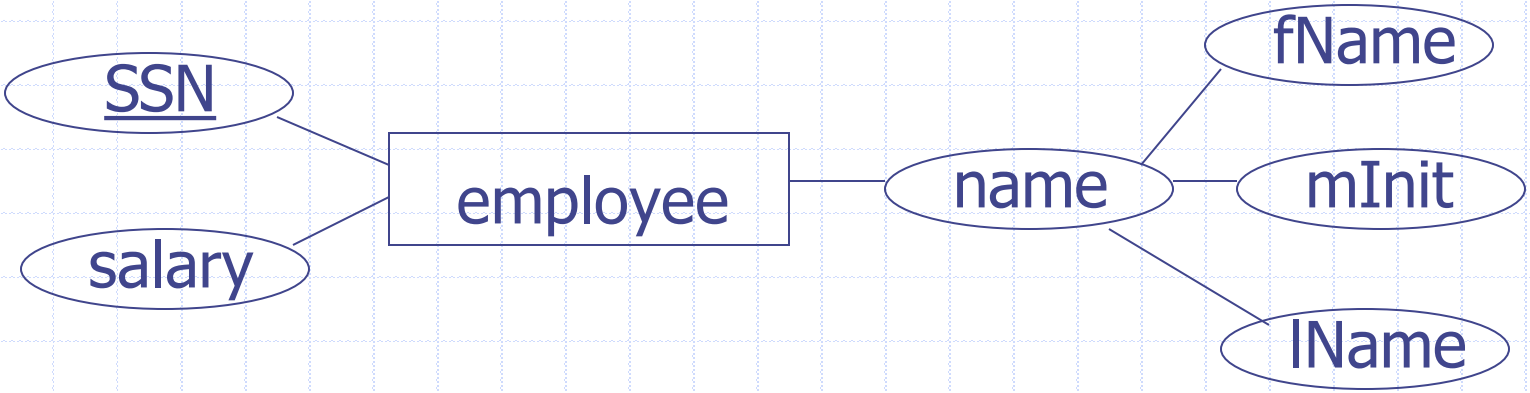
Ying Jin
Computer Science Department
California state University, Sacramento

# EER to Relational Mapping

- ERR is a conceptual model
- Map conceptual model to representational /implementation model

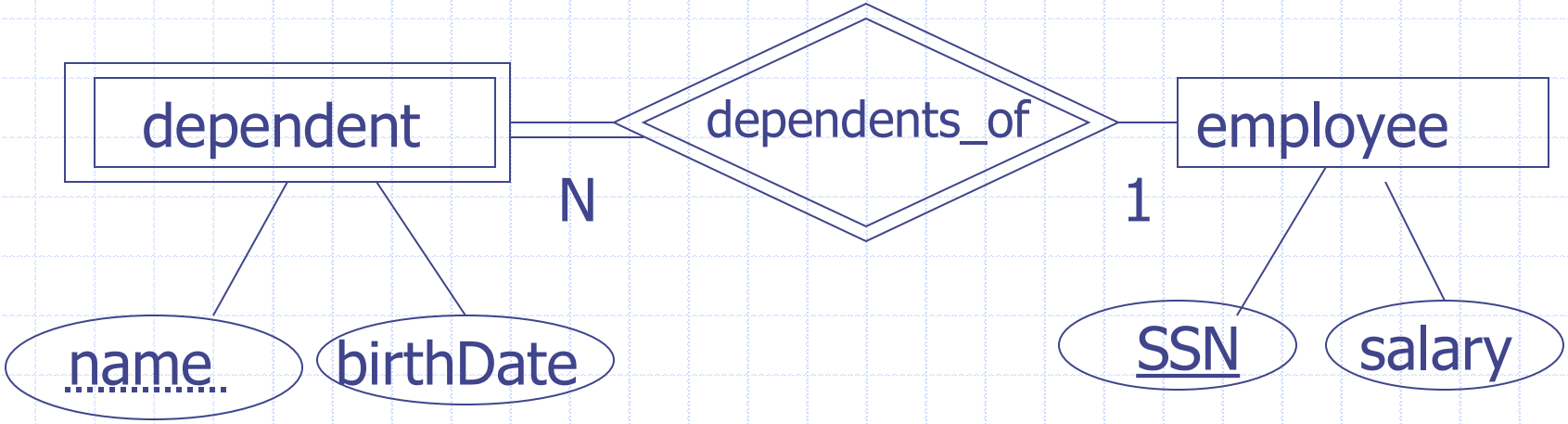
# (Strong) Entity

- Create a relation for the entity
- Include all the simple attribute and simple component attributes of a composite attribute



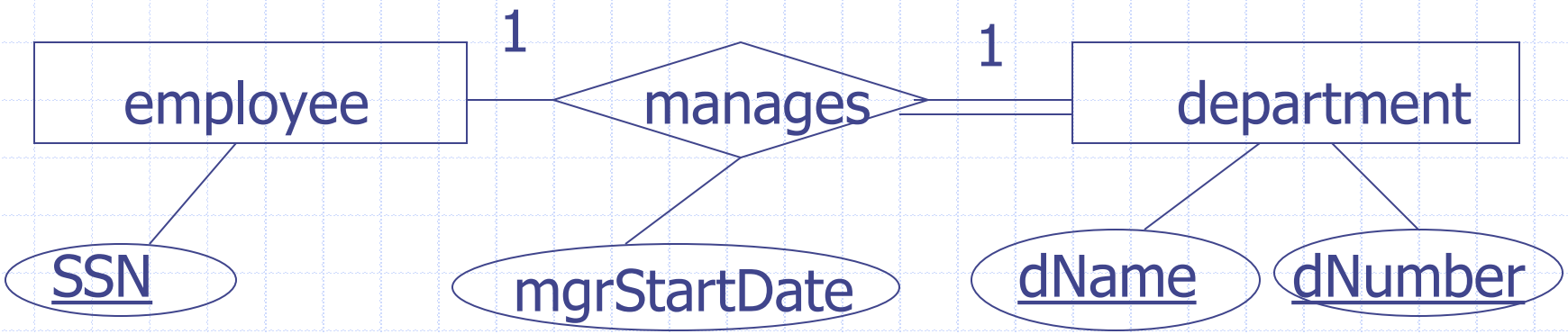
# Weak Entity

- The primary key of an weak entity is the combination of the primary key of the owner(s) and the partial key of the weak entity.



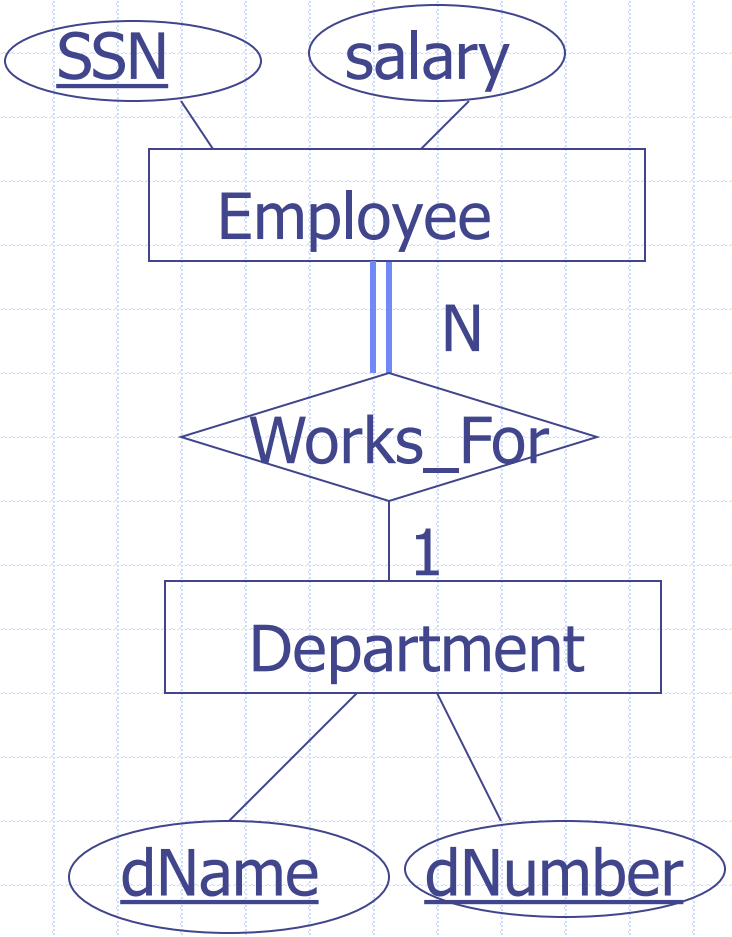
# 1:1 Relationship

- Chose one entity to include the primary key of the other entity as *foreign key*.
- Include simple attributes of the relation
- It is better to choose an entity type with total participation.



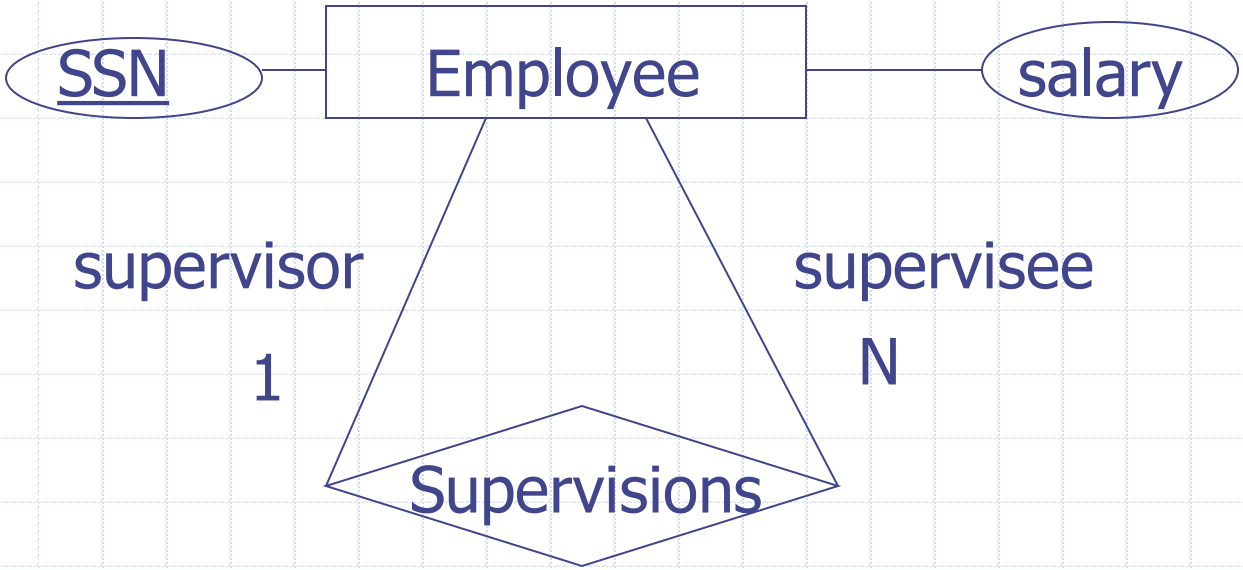
# 1:N Relationship

- The entity at N side includes the primary key of the entity at 1 side as foreign key.
- The entity at N side includes simple attributes of the relation



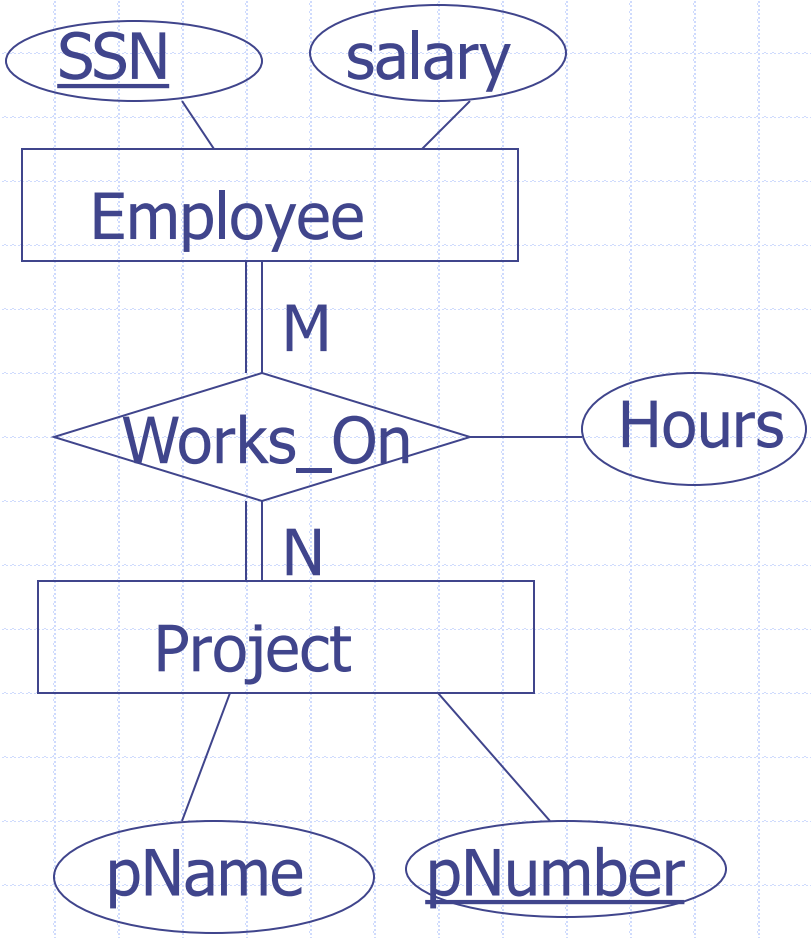
# 1:N Recursive Relationship

- The entity includes the primary key of itself as foreign key to represent the recursion



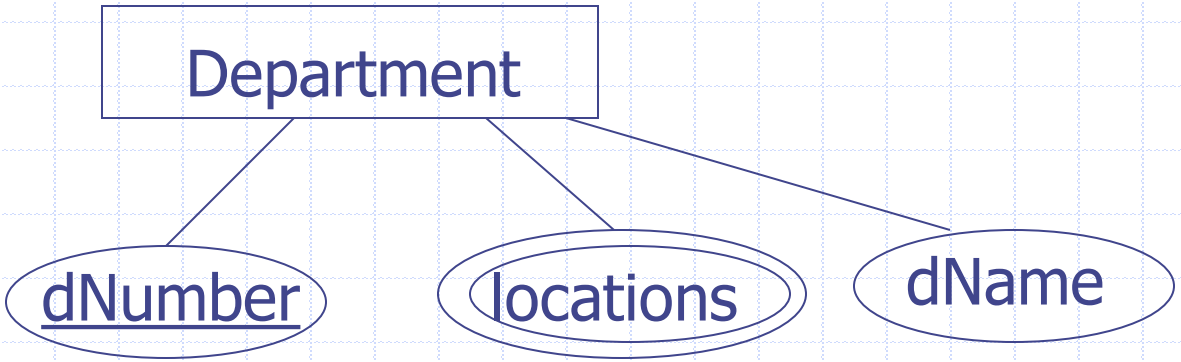
# M:N Relationship

- Create a new relation for the relationship
- Include the primary keys of the two entities as foreign keys.
- The combination of two foreign keys as the primary key of the new relation.



# Multivalued Attributes

- Create a new relation R for a multivalued attribute A (of entity E).
- R includes an attribute corresponding to A and the primary key of E as the foreign key.
- The combination of A and this foreign key as the primary key of R.

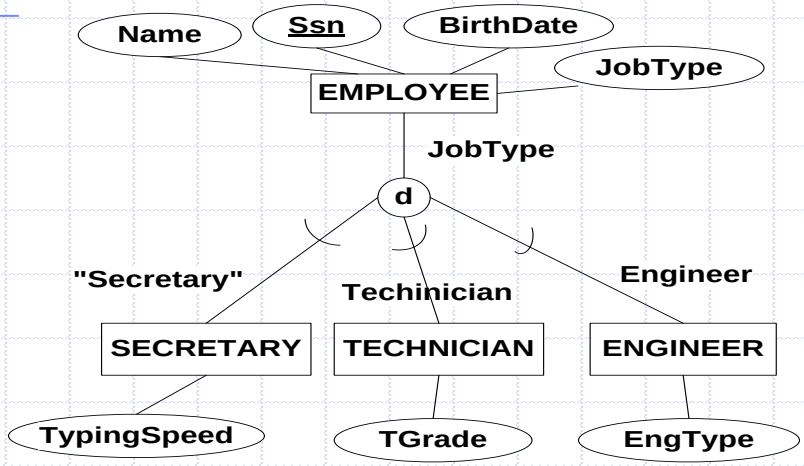


# Class Hierarchy

- Options for mapping class hierarchy
  1. Create a table for each class (both superclass and subclass).
  2. Create a table for each subclass only
  3. Flattening the hierarchy: create one table to present the class hierarchy, using type field to indicate superclass and subclass hierarchy.

# Class Hierarchy

## - 1. Create a table for each class

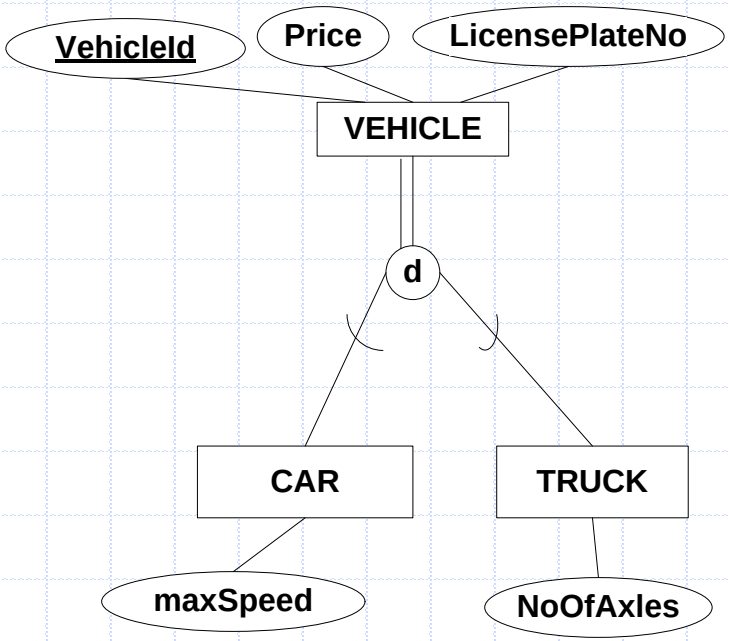



- This option works for any specialization (total or partial, disjoint or overlapping)
- Create a table for each class
- Create a view for each subclass

# Class Hierarchy

## - 2. Create a table for each subclass only

- This option only works for total and disjoint specialization
- Create a table for each subclass
- Create a view for the superclass



# Class Hierarchy

## - 3. Flattening the Hierarchy

- Create a single table for both the superclass and subclasses
  - Including all the attributes of the superclass and subclasses
  - Use type field to distinguish subclasses
  - If a tuple does not belong to a subclass, then the corresponding specific attributes of that subclass will have null values.
- Create a view for each superclass and subclass
- This option works for any specialization (total or partial, disjoint or overlapping)
- Not recommended if there are many specific attributes for the subclasses.

# 3.Flattening the Hierarchy
## - Different cases

- 3a) Attribute defined specialization
- 3b)Use single type attribute
- 3c) Use multiple type attributes

# Category

- Specify a *surrogate key*.
- Create a relation for the category class with the surrogate key
- Each superclass relation includes the surrogate key as a foreign key referencing the category class
- No associate views

# Category
## - Special Case

- A category whose superclasses have the same key, there is no need for a surrogate key.
- Example